*BIOSCIENCES BIOTECHNOLOGY RESEARCH ASIA*, December 2015. Vol. **12**(3), 2971-2981

# Evolutionary Synthesis of Large Discrete Systems with Dynamic Structure

**David Aregovich Petrosov[1], Vadim Alexandrovich Lomazov[1], Sergey Lgorevich Matorin[2], Alina Ivanovna Dobrunova[1] and Valentina Ivanovna Lomazova[2]**

[1]Belgorod State Agricultural University named after V. Gorin
Russia, 308503, Belgorod region, pos. Mayskiy, ul. Vavilova 1
[2]Belgorod National Research University
Russia, 308015, Belgorod, ul. Pobedy, 85.

DOI: http://dx.doi.org/10.13005/bbra/1981

(Received: 19 August 2015; accepted: 12 October 2015)

**The article discusses the issue of development of method for structural synthesis of large discrete systems with predetermined behavior and dynamic structure. This is aided by combined approach to development of structural synthesis based on simulation modeling, evolutionary methods and mathematical apparatus of Petri nets. The developed evolutionary procedures of system structural synthesis makes it possible to carry out structural synthesis in accordance with predetermined behavior not only by selection of components from elemental base but also by variation of inter-component couplings. The work proposes to apply genetic algorithms adapted to solution of problem of structural synthesis using nested Petri nets. The system component base applied for the synthesis procedure is expressed in the form of models based on Petri nets, the system models obtained by synthesis are estimated by computational experiment, that is, launching of the obtained system models by supplying of vector tuple to the model input and comparison of the tuple from the model output with reference one. Upon adaptation of genetic algorithm for operation with multi-component couplings it is proposed to highlight a separate layer of inter-component bus.**

**Key words:** System analysis, structural synthesis of large discrete systems, Evolutionary algorithms, genetic algorithm, Petri nets, simulation modeling.

Systems can be classified in accordance with the types of their state, behavior and identity. One of practically important classes of systems is large discrete systems (LDS). The systems of this class are characterized with significantly complex research due to their high dimensions. The systems of this class are comprised of separate modules (portions, elements, subsystems and so on), coupled between each other. Herewith, the number of portions and couplings is discrete (finite or countable), and internal processes in the object run under conditions of discrete time, which makes it possible to perform discrete model description. The considered class includes numerous social-economical, organizational-technological and engineering systems, such as power systems, elements and devices of computing tools and the like[1, 2].

Presently, recent researches imply development of new universal models and methods of structural and parametrical synthesis of LDS with predetermined behavior, which is urgent because the systems of this kind are widely used in various areas of interest. The essence of major issue encountered during solution of LDS synthesis is referred to this class of systems and is that at

* To whom all correspondence should be addressed.

existence of 10 components and about 10 instances of each component the number of possible configurations will equal to 10^10; herewith, it is ignored that the system can readjust during functioning due to dynamic alteration of inter-element couplings or alteration of operation parameters of its elements. These conditions can significantly complicate labor consuming synthesis of the system, that is, perform searching for system configuration which is able to transform predetermined input vector into required output one. Herewith, it should be mentioned that it is not always possible to vary the system elements, it would be more reasonable to deal only with the couplings between them.

## EXPERIMENTAL

### Simulation modeling

Taking into account labor consuming solution of such problems it is proposed to apply simulation modeling, since these tools make it possible to describe the processes in LDS as if they are real. The use of this method of mathematical simulation would facilitate simulation of synthesized system both for single and numerous tests.

In up-to-date simulation modeling the following mathematical tools are commonly applied:

- Differential equations;
- Finite state machines;
- Markov chains;
- Theory of graphs;
- Petri nets.

These theories are widely applied in the systems of simulation modeling and are used not only at the stages of LDS development, but during computation experiments with the models of existing objects and systems.

### Genetic algorithms

Application of only these tools does not eliminate high labor consumption upon solution of structural and parametric synthesis, and application of the methods based on random searching for solution is nearly impossible, thus, it is common to introduce the element of determinacy intrinsic for evolutionary methods. Among numerous evolutionary methods the most widely applied for solution of this class of problems are genetic algorithms (GA) and genetic programming. Application of these methods make it possible to reduce the time of both structural and parametrical synthesis of LDS, herewith, it is possible to solve not only the problem of system synthesis with fixed inter-component couplings but also with dynamic ones[3-6].

GA are based on the principles of natural selection and genetics, that is, the most promising specimens have the highest possibility to survive, in addition, they possess the properties of inheritance and can mutate.

A peculiar feature of GA with regard to other evolutionary methods is as follows:

1. GA operates with solutions which are expressed in the form of code line. The codes are transformed without any relation with their semantics.
2. Searching is based on the use of several points of solution space simultaneously. This eliminates possibility of undesired entering into local extremum of objective function, which is not unimodal.
3. Upon searching of GA only information about permissible values of parameters and objective function is used, which leads to significant speed increase.
4. In order to synthesize new points GA uses probability rules, and for transition from one points to another deterministic rules. Such merging of the rules is more efficient than their separate application[7-12].

It should be mentioned that this approach provides possibility to influence on searching for solutions by means of parameter presetting.

### Application of Petri nets for description of genetic algorithms

Genetic algorithms should be adapted to object region, where they will be applied, and, hence, it is required to select mathematical tools on the basis of which it will be possible not only to describe operation of adapted GA for solution of the problem of structural synthesis of LDS with fixed and dynamic inter-element couplings, but also to perform parametrical synthesis of the system.

The most suitable mathematical tool which satisfies the considered requirements is the theory of Petri nets. There is a wide range of variants of this tool: nested Petri nets, colored Petri nets, timed Petri nets, inhibitor Petri nets, marked Petri nets, and so on, that is, this tool makes it

possible to develop the model (depending on the object region it is required to select a class of Petri nets for simulation of elements) for any element of LDS[13-16]. Herewith, this mathematical tool makes it possible to combine two highlighted evolutionary methods: genetic programming and genetic algorithms. This is achieved due to the use of nesting properties (description of genetic algorithm at the upper level and expressing of marks of subsequent levels in the form of genotypes: models synthesized by LDS), and derivation of trees of achievable markings (tree encoding in genetic programming)[17].

## RESULTS

### Statement of problem

Let us consider the following class of problems of structural synthesis of LDS with dynamic structure.

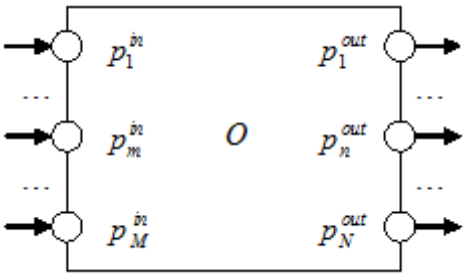



**Fig. 1.** Context model of LDS in the problem of structural synthesis

Given:

$$O = \left\langle In, Out, \{S_k\}_{k=1}^{K}, \{f_a\}_{a=1}^{A}, \{F_b\}_{b=1}^{B} \right\rangle,$$

where *O* is the LDS, structural synthesis of which should be performed;
*In* is the set of input data of O-LDS;
*Out* is the set of output data of O-LDS;
*Sk* is the k[th] subsystem of O-LDS;
¦a is the function, determining the input data corresponding to input data: ¦a : *In*®*Out*;
Fb is the binary ratio at the set {Sk}Kk=1 : Fb Ì{Sk}Kk=1´{Sk}Kk=1 .

For the given function $f_{a_0}$ it is required to select binary ratio $F_{b_0}$ so that the set of subsystems {Sk}Kk = 1 would provide processing of input data by the O system in accordance with the function $f_{a_0}$.

### Formalization using Petri nets

Inputs of O-LDS will be simulated by the set of positions $P_{in} = \{p_m^{in}\}_{m=1}^{M}$, where $M$ is the number of system inputs, and its outputs by the set of positions $P_{out} = \{p_n^{out}\}_{n=1}^{N}$, where $N$ is the number of system outputs (Fig. 1).

Inputs and outputs of O-system will also be simulated by a set of positions. Inputs of the *Sk*-subsystem are the set $P_k^{in} = \{p_{k,m}^{in}\}_{m=1}^{M(k)}$, where M(k) is the number of inputs of the *Sk*-subsystem, and its outputs are the set $P_k^{out} = \{p_{k,n}^{out}\}_{n=1}^{N(k)}$, where N(k) is the number of outputs of the Sk-subsystem (Fig. 2).

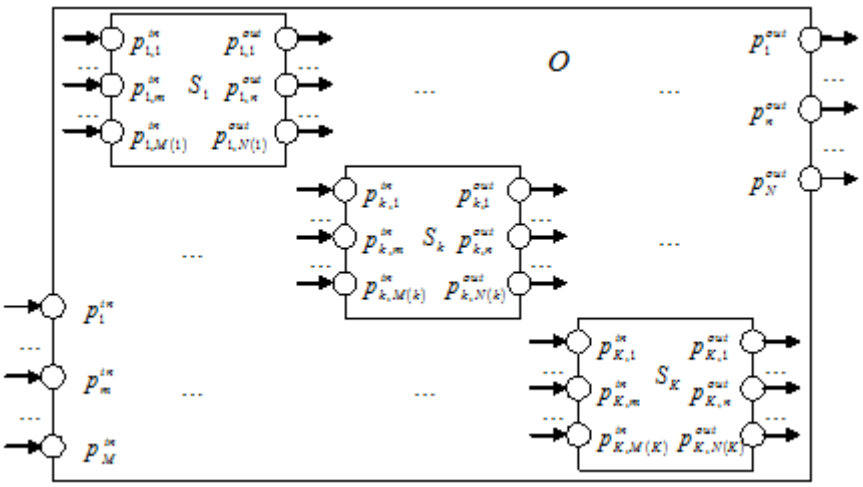



**Fig. 2.** Multiple subsystems (elements) of O LDS

Subsystems $\{S_k\}_{k=1}^{K}$ can be coupled both between each other and with the inputs and outputs of O-system. These couplings will be simulated by the set of transitions $T = \{t_q\}_{q=1}^{Q}$, where Q is the number of transitions. Each *tq* transition is characterized by its input and output positions. Inputs of *tq* transition can be both any inputs of O-system and any outputs of subsystems $\{S_k\}_{k=1}^{K}$. Outputs of *tq* outputs can be both outputs of O-system and any inputs of subsystems $\{S_k\}_{k=1}^{K}$. Let us denote the inputs of *tq* transition as *Inq*, and the outputs as *Outq*. Then, on the basis of the aforementioned, $In_q \subset P_{in} \bigcup \left( \bigcup_{k=1}^{K} P_k^{out} \right)$ è $Out_q \subset P_{out} \bigcup \left( \bigcup_{k=1}^{K} P_k^{in} \right)$.

It is quite natural that for each input of O-system a *tq* transition should exist coupled with this input. In addition, for each output of each subsystem *Sk* a *tq* transition should exist coupled with this output. Formally, these requirements can be written in the form of equality $\bigcup_{q=1}^{Q} In_q = P_{in} \bigcup \left( \bigcup_{k=1}^{K} P_k^{out} \right)$.

Similarly, for each O-LDS output a *tq* transition should exist coupled with this output, and for each subsystem *Sk* a *tq* transition should exist coupled with this input. Formally, these requirements can be written in the form of equality $\bigcup_{q=1}^{Q} Out_q = P_{out} \bigcup \left( \bigcup_{k=1}^{K} P_k^{in} \right)$.

Different transitions can have certain common input and output position, but there should not be totally coinciding transitions. This requirement is formally written as follows:

$In_{q_1} = In_{q_2} \Rightarrow Out_{q_1} \neq Out_{q_2}$ (ïðè $q_1 \neq q_2$);

$Out_{q_1} = Out_{q_2} \Rightarrow In_{q_1} \neq In_{q_2}$ (ïðè $q_1 \neq q_2$).

The set of subsystems together

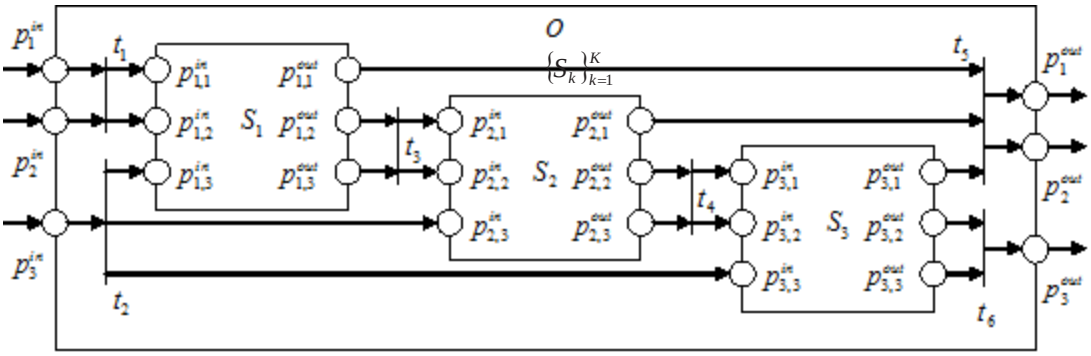



**Fig. 3.** An example of system structure

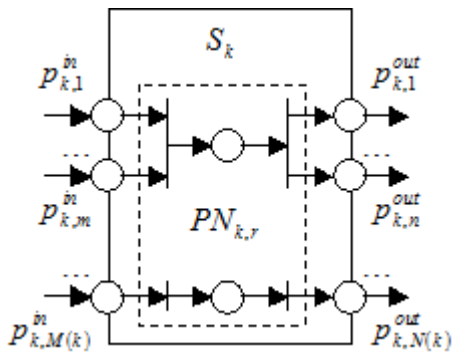



**Fig. 4.** Expressing algorithm of processing by Petri net

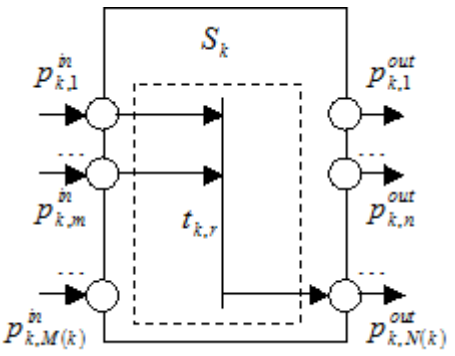



**Fig. 5.** Expressing of processing algorithm by transition

with the subset of transition set $T = \{t_q\}_{q=1}^{Q}$ completely determine current configuration of O-system. Not all subsystems $\{S_k\}_{k=1}^{K}$ should be involved in a current configuration. Herewith, rearrangement of configuration (synthesis) of O-system is determined by variation of subset of currently active transitions. An example of system configuration is illustrated in Fig. 3.

Rearrangement of functions (variation of functioning algorithm of O-system) occurs in subsystems $\{S_k\}_{k=1}^{K}$. In general case each subsystem *Sk* can be correlated with a set of Petri nets, which will be the models of data processing programs. Let us denote this set as *PNk* and define it as follows: $PN_k = \{PN_{k,r}\}_{r=1}^{R(k)}$, where *PNk,r* is the r[th] algorithm of data processing by the subsystem *Sk*, expressed in the form of Petri net (Fig. 4).

It is formally required for each Petri net *PNk,r* to describe its positions, transitions and arcs. But in general case the net *PNk,r* simulates certain action on transformation of input data into output data. Thus, for the sake of simplicity, not Petri nets *PNk,r* will be considered as algorithm

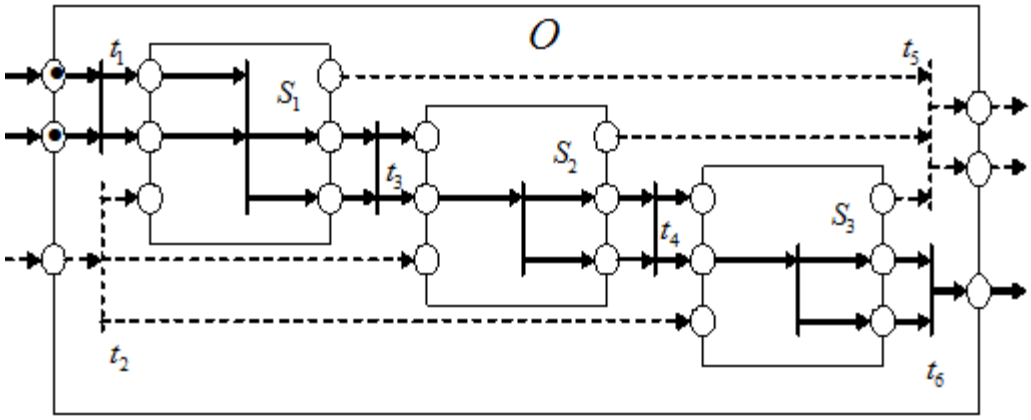



**Fig. 6.** Example of O-system model

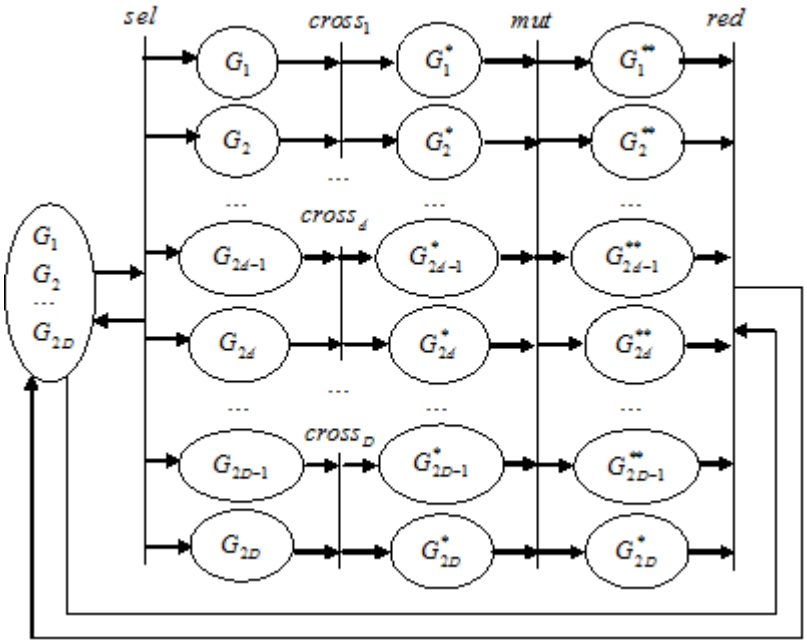



**Fig. 7.** Description of genetic algorithm by Petri net

models, but transitions ("degenerate" Petri nets). Therefore, the set *PNk* will be expressed as $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$. Inputs *Ink,r* of each transition *tk,r* can be any inputs of subsystem *Sk*: $In_{k,r} \subset P_k^{in}$, and outputs *Outk,r* of each transition *tk,r* can be any outputs of subsystem *Sk*: $Out_{k,r} \subset P_k^{out}$ (Fig. 5.).

On the basis of the aforementioned a model of O-system in the form of Petri net can expressed, for instance, as illustrated in Fig. 6. In this model the bold line highlights the route along which the input data, supplied to upper two inputs of O-system (highlighted by two marks in the figure), pass across its subsystems and appear at the lower output of O-system. Dashed line highlights inactive transitions and arcs.

**Description of genotype**

GA should be adapted to actual object region by preset of parameters and determination of GA operators. When using GA, alternative solutions are expressed in the form of symbol line of fixed length and referred to as genotype. Thus, special attention should be paid to formal description of genotype.

In our case genotype G should define::

- Which transitions of the set $T = \{t_q\}_{q=1}^{Q}$ will be expressed in the O-system model;
- Which transactions of the set $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$ will be expressed in the model of each subsystem *Sk*.

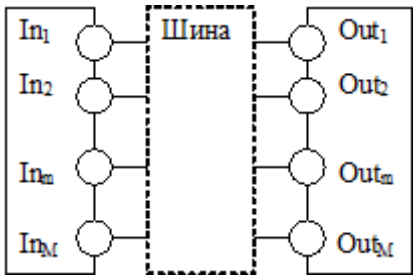



**Fig. 8.** Generalization of inter-component couplings into "Bus" layer

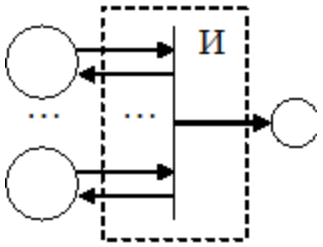



**Fig. 9.** "AND" element

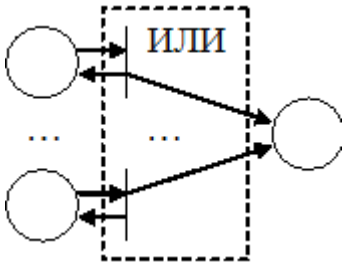



**Fig. 10.** "OR" element

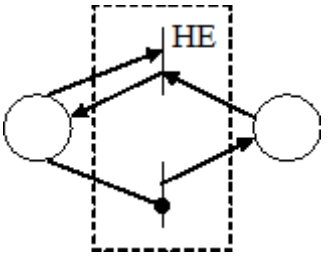
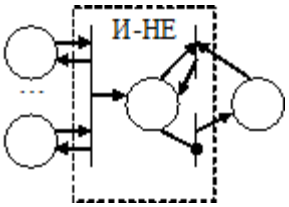



**Fig. 11.** "NOT" element



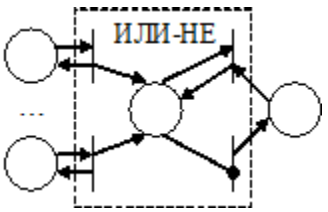




**Fig. 12.** "NAND" and "NOR" elements

Therefore, genotype G will be as follows: G = (g1,…, gq,…gQ, h1, …, hk,…,hK), where *gq* Î{0,1} and *hk* Î{0,1,…,*r*,…,*R(k)*}..

If *gq* = 0, then the *tq* transition is absent in the model of Oobject, otherwise (*gq* = 1) the *tq* transition is present in the O-system model. If *hk* = 0, then the model of subsystem *Sk* has no algorithm of data processing (the program is not loaded). If *hk* = *r* (*r*Î{1,…,*R(k)*}), then the model of subsystem *Sk* contains *tk,r* transition.

Therefore, the number $C$ of all hypothetically possible models of O-system is determined as follows:

$$C = 2^{Q}\prod_{k=1}^{K}(R(k)+1).$$

Hence, even at minor amount of couplings, subsystems and algorithms of data processing the number of O-system models is very high. For instance, at *Q* = 10, *K* = 10 and *R*(1) = *R*(2)=…=*R*(10) we have *C* = 210´1110 = 2210.

**Objective function**

Among all hypothetically possible models of O-system we should determine such, which solves the given problem: on the basis of available set of subsystems $\{S_k\}_{k=1}^{K}$ and transition system $T = \{t_q\}_{q=1}^{Q}$ to construct such O-system which will react to input vector (or a set of vectors) by required output vector (or a set of vectors). Thus, the closer is the output vector to the required one, the better is the system obtained by synthesis.

In order to estimate this closeness let us introduce notation for the required vector: $V = (v_1,\ldots,v_n,\ldots v_N)$, where *vn* is the number of marks which should be in the position $p_n^{out}$ of O-system. It is natural that *vn*Î{0,1,2,3,…}. The vector obtained as a result of operation of O-system we denote as *W* = *(w1,…,wn,…wN*), where *wn* is the number of marks which in fact are in the position $p_n^{out}$ of O-system. Similarly, *wn* Î {0,1,2,3,…}..

The difference (distance) between the required and actual vector can be estimated, for instance, by the equation: $\rho(V,W) = \sum_{n=1}^{N}|v_n - w_n|$.

Selection of objective function influences on efficiency of operation of genetic algorithm. Thus, in practice it is possible to apply the equation setting classical Euclidian distance $\rho(V,W) = \sqrt{\sum_{n=1}^{N}(v_n - w_n)^2}$, and rarely applied equation $\rho(V,W) = \max_{1\le n\le N}|v_n - w_n|$. Upon software implementation of genetic algorithm it would be reasonable to include the possibility of operation with equation $\rho(V,W) = \left(\sum_{n=1}^{N}|v_n - w_n|^p\right)^{1/p}$ into the system, which at *p* = 1, *p* = 2 and *p* '! " expresses all equations considered above. Varying the parameter *p*, it is possible to influence on the efficiency of operation of genetic algorithm.

The lower is $\rho$, the closer is the O-system obtained by synthesis to the required one. At the system completely corresponds to the preset requirement. Nevertheless, there is no guarantee that it is possible to construct the required O-system on the basis of a given set of subsystems , their models $PN_k = \{t_{k,r}\}_{r=1}^{R(k)}$ and (co)plings $T = \{t_q\}_{q=1}^{Q}$.

**Operators of genetic algorithms**

After selection of object function it is possible to describe operators of genetic algorithm.

Selection operator should select those genotypes for crossing, which are the closest to the required configuration. This process can be arranged, for instance, as follows: to order all available genotypes in terms of quality (best to worst) and to cross neighboring pairs. It is possible to allow the best genotypes to participate in several crossings.

Crossing operator can be described as follows. Let us consider two genotypes: $G_1 = (g_1^1,\ldots,g_q^1,g_{q+1}^1,\ldots g_Q^1,h_1^1,\ldots,h_k^1,h_{k+1}^1,\ldots,h_K^1)$ and $G_2 = (g_1^2,\ldots,g_q^2,g_{q+1}^2,\ldots g_Q^2,h_1^2,\ldots,h_k^2,h_{k+1}^2,\ldots,h_K^2)$

We select at random two numbers: $q$ of the range (set) {1,2,…,*Q*} and *k* of the range {1,2,…,*K*}. And then we modify the respective portions of the genotypes. Therefore, the parents *G1* and *G2* provide the descendants *G3* and *G4*,

inheriting the properties of the parents:

$$G_3 = \left(g_1^1,\ldots,g_q^1,g_{q+1}^2,\ldots g_Q^2,h_1^1,\ldots,h_k^1,h_{k+1}^2,\ldots,h_K^2\right)$$

$$G_4 = \left(g_1^2,\ldots,g_q^2,g_{q+1}^1,\ldots g_Q^1,h_1^2,\ldots,h_k^2,h_{k+1}^1,\ldots,h_K^1\right)$$

It is possible to be limited with selection of only one number (*q* or *k*). How many numbers will be selected and if one, then which exactly — it can also be determined at random. It is possible to select two points for each range ({1,2,…,*Q*} and {1,2,…,*K*}) and to exchange with "middles".

Mutation operator can be described as follows. Let us consider one genotype *G=(g1,…,gq,…gQ,h1,…,hk,…,hK)*. Let us select at random two numbers: *q* of the range {1,2,…,*Q*} and *k* of the range {1,2,…,*K*}. Then we change the values *gq* and *hk* as follows. If *gq*=0, then we reverse it to unity: *gq* = 1. And vice versa, if we had *gq* = 1, then we have *gq* = 0. If we had *hk* = *r*, then we replace it with any other of the range {0,1,…,*r-1,r+1,…,R(k)*}. Naturally, at *r* = 0 it will be the range {1,…,*R(k)*}, and at *r* = *R(k)* it will be the range {0,1,…,*R(k)*-1}.

Reduction operator should remove weak genotypes. With this aim after crossing it is necessary to detect the properties of all descendants. Then, the descendants and the parents should be combined into one set of genotypes and be ordered in terms of quality from best to worst. Those in the worst segment should be eliminated.

**Solution of the problem**

Prior to initiation of operators of genetic algorithm an initial population of genotypes should be preset. If there are any hypotheses on the structure and functioning of O-system, then the genotypes can be described by the designer. Otherwise, it is possible to generate several genotype at random. The size of population is also determine dby the designer.

Operation of genetic algorithm can also be described by Petri net (Fig. 7) as in previous works [1, 17].

Initial population of genotypes *G1,G2,…G2D* is positioned in Petri net. Operator *SEL* performs selection of genotypes for crossing. Operators *CROSS1,…,CROSSd,…,CROSSD* perform crossing of genotypes which then are exposed to mutation by operator *MUT*. The cycle is finished by operator *RED*, which eliminates weak genotypes. Then the process is repeated.

**Model of inter-component bus**

In a particular case rearrangement of O-system structure can be carried out by means of inter-component bus (IB). This method is widely applied in LDS synthesis and requires individual discussion [18-20].

Overall system of LDS elements and buses is processed sequentially from the system input to output. Initially the first layer of elements receives and processes input signals. Then, using the first bus, the signals are transferred to the second layer. After processing the signals via the second bus are transferred to the third layer and so on. The process is finished after processing of the signals by the last layer of elements, and the output vector is the required one [21, 22].

Formally, for each bus a set of initial positions *In = {Inm}* and a set of output positions *Out = {Outm}* (*m* = 1, 2, …, *M*) are predetermined. The input positions are the outputs of a previous block of the system elements, and the output positions are the inputs of a previous block of elements (Fig. 8).

In order to simulate IB it would be reasonable to use both elements selected for synthesis, and logical elements. Each output position *Outm* is associated (randomly generated) with:

– "OR" block with identification of the set of input positions;
– "NOR " block with identification of the set of input positions;
– the number of "AND" blocks;
– the number of "NAND " blocks;
– the set of initial positions for each "AND" ("NAND") block.

Output position can be associated only with one "OR" block, one "NOR " block, several "AND" blocks, several "NAND" blocks.

Bus operates as follows. Output positions are processed sequentially from *Out1* to *OutM*, according to the algorithm. At first "OR" block is processed, then "NOR" block. Further, "AND" blocks are checked and, at the end, "NAND" blocks. In order to accelerate the procedure the "AND" blocks are initially checked with lower number of input positions. The check is stopped when a block is found which can be launched.

Therefore, a bus is completely characterized with the set of blocks associated with

each output position. Thus, crossing of buses is the exchange with output positions.

Parents:

X-bus: *xOut1, …, xOutm, xOutm+1, …, xOutM*

Y-bus: *yOut1, …, yOutm, yOutm+1, …, yOutM*

Descendants:

X-bus: *xOut1, …, xOutm, yOutm+1, …, yOutM*

Y-bus: *yOut1, …, yOutm, xOutm+1, …, xO*utM

Mutation of bus is modification for any of output position:

– the set of input positions of "OR" block;

– the number of "AND" blocks;

– the set of input positions for each "AND" block.

The "AND" element is simulated by transition, the inputs of which are coupled with the positions from which signals are supplied, and the inputs — with the position to which they are supplied (Fig. 9).

The "OR" element is simulated by a block of transitions, the inputs of which are coupled with the positions from which signals are supplied, and the inputs — with the position to which they are supplied (Fig. 10).

The block operates according to the following rule. Transitions are polled top-down. If any transition can operate, then it is initiated and the other transitions are not polled.

The "NOT" element is simulated by two transitions, the inputs of which are coupled with position, the signal from which should be inverted, and the outputs with position to which the inverted signal is supplied (Fig. 11).

The element operates as follows. Single signal at input enables removal of a token from input position. The signal absence at input does not restrain the lower transition, which adds token to the input position.

On the basis of models of "AND", "OR" and "NOT" elements it is possible to construct models of "NAND" and "NOR " elements (Fig. 12).

It should be noted that the "NAND" and "NOR" blocks with one input are not equivalent to the "NOT" element.

## DISCUSSION

In order to initiate operation of the proposed model it is required to position the models of element base, which can be used in the system synthesis, at origination of the expressed net. The SEL transition generates initial population of the models of synthesized system, estimates the degree of approximation of the obtained results to the required solution (by means of computation experiment) and prepares pairs for the CROSS transition. The CROSS transition performs the function of mating of parent pairs, and then the descendants will be preserved. The MUT transition performs mutation of certain specimens. New generation of specimens is verified for conformity to preset criteria by means of computation experiment, that is, the obtained by synthesis models based on Petri nets initiate their operation by supplying of a set of vectors to input and are compared with preset reference vectors at the model output by determination of the distance between them in Cartesian coordinates. Thus, the quality of the obtained by synthesis LDS is estimated. The most remote from the required solutions specimens will be eliminated by the RED operator, and the most adapted ones (closer to the synthesis conditions) will continue searching until the shutdown condition of operation of the proposed model is reached. Such conditions are as follows:

- Determination of model of synthesized large discrete system with total conformity to the synthesis criteria;
- Termination of synthesis time of large discrete system (in this case the system models will be proposed, which are the closest to the synthesis criteria);
- Limitation in number of processed populations (in this case the system models will be also proposed, which are the closest to the synthesis criteria).

It should be mentioned that the proposed model of genetic algorithm formalized by nested Petri nets provides possibility to work not only with the models of elements but also with couplings between them. In this case the SEL, CROSS and MUT do not substitute the elements of synthesized system but operate with inter-element coupling, that is, modify the couplings between the system elements. This is aided by dedicated layers of inter-component buses.

## CONCLUSIONS

The main result of this work is the

development of genetic algorithm model, which facilitated formalization and algorithmization of structural synthesis of large discrete systems with predetermined behavior by generation of solution with dynamic structure on the basis of preset components.

Solution of this problem is significantly complicated by huge amount of possible variants of implementation, from which it is necessary to select those that satisfy the preset behavior. Three theories were applied in solution of this problem: evolutionary methods, simulation modeling, and Petri nets. In order to eliminate this difficulty genetic algorithms were applied adapted to solution of the given problem by using multi-level nested Petri nets.

In order to solve the complicated problem in terms of computation load a separate layer of inter-element couplings was applied: "inter-component bus". This enabled optimization of the system synthesis using genetic algorithm based on nested Petri nets.

The proposed approach will be further applied to the problem of parametric synthesis of large discrete systems with preset behavior with dynamic and static inter-component couplings, which implies operation of genetic algorithm not only with components of synthesized system and couplings between them, but also with adjustment of components of large discrete system. This will facilitate creation of universal method of structural and parametric synthesis of large discrete systems. Solution of the problem by existing methods at present is labor consuming, since the searching area is large.

In theoretical aspect it seems to be promising to develop the proposed approach in the problems of development of universal methods of structural and parametric synthesis of large discrete systems with modifying structure and composition of involved components.

## ACKNOWLEDGMENTS

The work was supported by Russian Foundation for basic researches (projects Nos. 14-07-00246, 15-07-02371).

## REFERENCES

1. Lomazov V.A., Lomazova V.I., and Petrosov D.A. Evolutionary Support of Decision Making in Simulation of Interrelated Processes // Issues of Up-to-date Science and Practice. Vernadsky University. 2014. No. 2 (51). PP. 82-89.
2. Semenkin Å.S. Evolutionary strategies algorithm based approaches for the linear dynamic system identification // Adaptive and Natural Computing Algorithms. Lecture Notes in Computer Science, Volume 7824. – Springer-Verlag, Berlin, Heidelberg, 2013. - P. 477-484.
3. Emelyanov V. V., Kureichik V. V., and Kureichik V. M. Theory and Practice of Evolutionary Simulation. — Moscow: Fizmatlit, 2003. — P. 432
4. Paolo Sibani & Henrik Jeldtoft Jensen (2013). Stochastic Dynamics of Complex Systems, ISBN 978-1-84816-993-7, World Scientific and Imperial College Press. 300pp.
5. Kureichik V. M., Lebedev B. K., and Lebedev O. K. Search Adaptation: Theory and Practice. — Moscow: Fizmatlit, 2006. — P. 272.
6. Gladkov L. A., Kureichik V. V., and Kureichik V. M. Genetic Algorithms: Textbook. — 2 edition — Moscow: Fizmatlit, 2006. — P. 320.
7. Eiben, A. E. et al (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87.
8. Hingston, Philip; Barone, Luigi; and Michalewicz, Zbigniew (2008). Design by Evolution: Advances in Evolutionary Design. Springer. ISBN 978-3540741091.
9. Schmitt, Lothar M (2001), Theory of Genetic Algorithms, Theoretical Computer Science 259: 1–61
10. Schmitt, Lothar M (2004), Theory of Genetic Algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling, Theoretical Computer Science 310: 181–231.
11. Akbari, Ziarati (2010). "A multilevel evolutionary algorithm for optimizing numerical functions" IJIEC 2 (2011): 419–430
12. Poli, R., Langdon, W. B., McPhee, N. F. A Field Guide to Genetic Programming.- Lulu.com, freely available from the internet, 2008. -ISBN 978-1-4092-0073-4. 250pp.

13. Peterson G. Theory of Petri Nets and Simulation of Systems. — Moscow: Mir, 1984. — 264 p.
14. Kotov V. E. Petri Nets. — Moscow: Nauka, 1984. — 160 p.
15. Dawis, E. P., J. F. Dawis, and Wei-Pin Koo (2001). Architecture of Computer-based Systems using Dualistic Petri Nets. Systems, Man, and Cybernetics, 2001 IEEE International Conference on Volume 3, 2001 Page(s):1554 - 1558 vol.3
16. V. Z. Magergut and A. A. Klimov Adaptive Approach to Development of Algorithms of Operation of Logical Devices Marked by Petri Nets // Proceedings of XXIII International Scientific Conference "Mathematical Methods in Engineering and Technologies (ÌÌÒÒ-23)". Vol 10. Saratov: SGTU, 2010. – PP. 17–20.
17. Petrosov, D.A Lager discrete systems evolutionary synthesis procedure/ Petrosov, D.A., Lomazov V.A., Lomazova V.I., Matorin S.I. and Dobrunova A.I// Biosciences biotechnology research Asia Vol. 12(2), 2015., 1767-1775P..
18. Marakhovskiy V. B., Rozenblyum L. Ya., and Yakovlev A. V. Simulation of Parallel Processes. Petri Nets. Course for System Architects, Programmers, System Analysts, Designers of Complex Control Systems. - St. Petersburg: Professional Literature, IT Training, 2014. - 400 p.
19. V. A. Ignatenko and V. Z. Magergut. Petri Information Net as Basis of Fuzzy Control of Objects // Proceedings of XXIII International Scientific Conference "Mathematical Methods in Engineering and Technologies (ÌÌÒÒ-23)". Vol 10. Saratov: SGTU, 2010. – PP. 13–17.
20. V. A. Ignatenko and V. Z. Magergut. Parallel Processing of Control Algorithms Using Petri Information Net // Mathematical Methods in Engineering and Technologies: Proceedings of XXIV International Scientific Conference in 10 volumes. Section 6,7; Editor: V. S. Balakirev. – Kiev: National Technical University of Ukraine "KPI", 2011. – Vol. 6. – PP. 73–75.
21. V. A. Ignatenko and V. Z. Magergut. Petri Information Net as a Tool for Parallel Processing of Control Algorithms // Scientific journal of BelGU. History, Political science, Economy, Informatics. 2011. Issue No. 19. – PP. 119–126.
22. V. A. Ignatenko and V. Z. Magergut. Description of Dynamic Processes Using Petri Information Net // Scientific journal of BelGU. History, Political science, Economy, Informatics. – 2011. – Issue No. 13. – PP. 161–179